

Objectives

- Describe packages and list their possible components
- Create a package to group together related variables, cursors, constants, exceptions, procedures, and functions
- Designate a package construct as either public or private
- Invoke a package construct
- Describe a use for a bodiless package

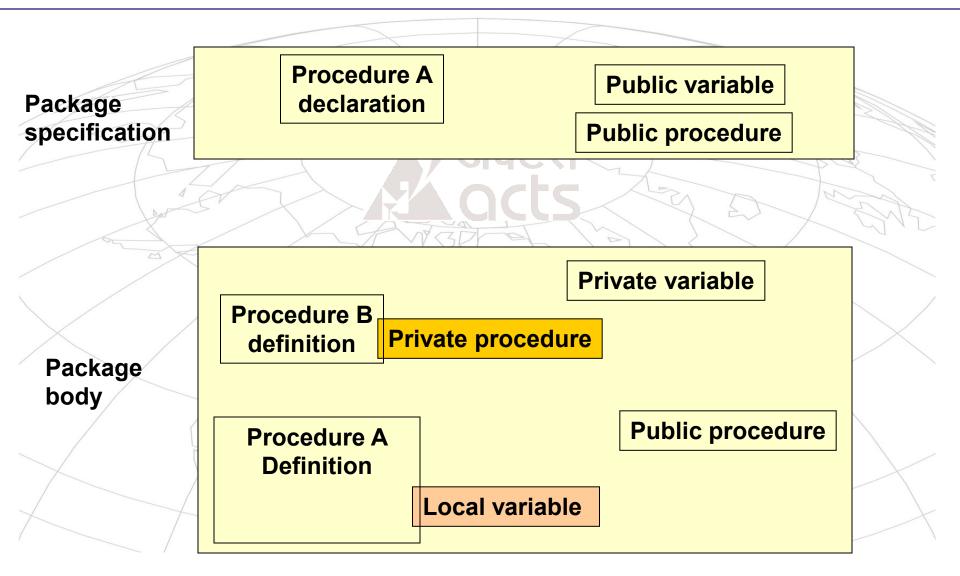


Overview of Packages

- Packages:
 - Group logically related PL/SQL types, items, and subprograms
- Consist of two parts:
 Specification
 Body
- Cannot be invoked, parameterized, or nested
- Allow the Oracle server to read multiple objects into memory at once



Components of a Package





Creating the Package Specification

CREATE [OR REPLACE] PACKAGE

package_name
IS|AS

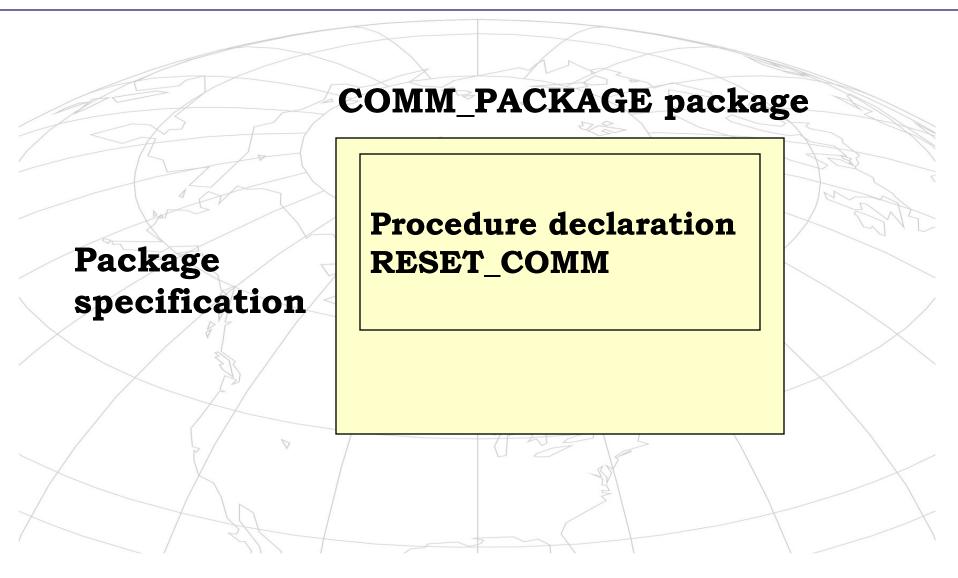
public type and item declarations

subprogram specifications
END package_name;

- The REPLACE option drops and recreates the package specification.
- Variables declared in the package specification are initialized to NULL by default.
- All the constructs declared in a package specification are visible to users who are granted privileges on the package



Declaring Public Constructs





Creating a Package Specification: Example

```
CREATE OR REPLACE PACKAGE comm_package IS

g_comm NUMBER := 0.10; --initialized to 0.10

PROCEDURE reset_comm

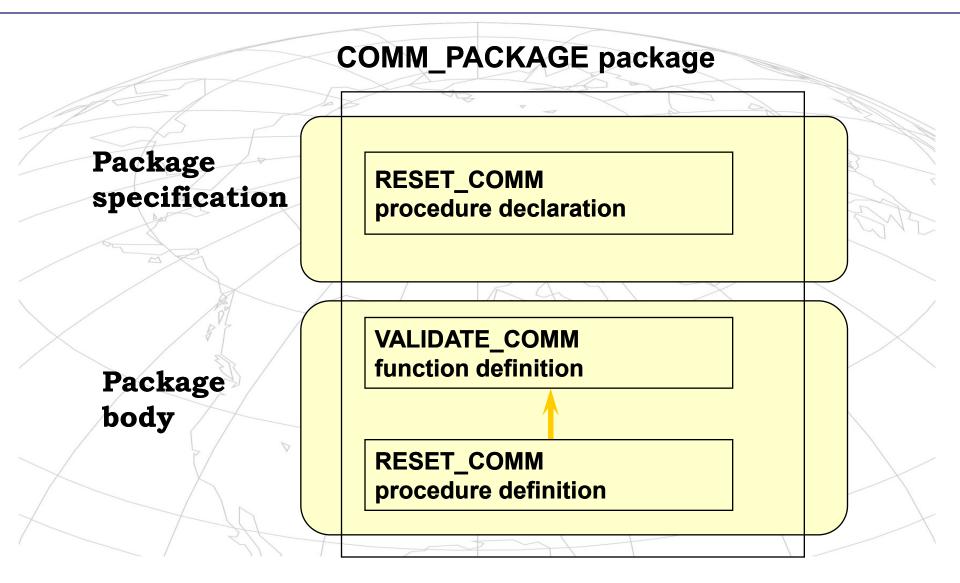
(p_comm IN NUMBER);

END comm_package;
/
```

- G_COMM is a global variable and is initialized to 0.10.
- RESET_COMM is a public procedure that is implemented in the package body.



Public and Private Constructs





Creating a Package Body: Example

```
CREATE OR REPLACE PACKAGE BODY comm_package
IS
   FUNCTION validate_comm (p_comm IN NUMBER)
   RETURN BOOLEAN
   IS
   v_max_comm NUMBER;
  BEGIN
   SELECT MAX(comm)
   INTO v_max_comm
   FROM emp;
   IF p_comm > v_max_comm THEN RETURN(FALSE);
      ELSE RETURN(TRUE);
   END IF;
  END validate_comm;
```



Creating a Package Body: Example

```
PROCEDURE reset_comm (p_comm IN NUMBER)
 IS
 BEGIN
 IF validate_comm(p_comm)
 THEN g_comm:=p_comm; --reset global variable
 ELSE
 RAISE APPLICATION ERROR(-20210, 'Invalid
  commission');
 END IF;
 END reset_comm;
END comm_package;
```



Invoking Package Constructs

Example: Invoke a function from a procedure within the same package.

```
CREATE OR REPLACE PACKAGE BODY comm_package
IS
  PROCEDURE reset_comm
  (p_comm IN NUMBER)
  IS
  BEGIN
  IF validate_comm(p_comm)
  THEN g_comm := p_comm;
  ELSE
  RAISE_APPLICATION_ERROR
  (-20210, 'Invalid commission');
  END IF;
  END reset_comm;
END comm_package;
```



Declaring a Bodiless Package

```
CREATE OR REPLACE PACKAGE global_vars IS
  mile 2 kilo CONSTANT NUMBER := 1.6093;
  kilo_2_mile CONSTANT NUMBER := 0.6214;
  yard_2_meter CONSTANT NUMBER := 0.9144;
  meter_2_yard CONSTANT NUMBER := 1.0936;
END global_vars;
EXECUTE DBMS_OUTPUT.PUT_LINE('20 miles = '| |20*
global_vars.mile_2_kilo | | km')
```



Referencing a Public Variable from a Procedure

```
CREATE OR REPLACE PROCEDURE meter_to_yard
(p_meter IN NUMBER, p_yard OUT NUMBER)
IS
BEGIN
 p_yard := p_meter * global_vars.meter_2_yard;
END meter_to_yard;
declare
yard NUMBER := 1;
begin
```

meter_to_yard (1, yard);
Copyright © CDAC-ACTS



Removing Packages

To remove the package specification and the body, use the following syntax:

DROP PACKAGE package_name;

To remove the package body, use the following syntax:

DROP PACKAGE BODY package_name;



Guidelines for Developing Packages

- Construct packages for general use.
- Define the package specification before the body.
- The package specification should contain only those constructs that you want to be public.
- The package specification should contain as few constructs as possible.

Also avoid writing packages that duplicate features provided by Oracle.



Advantages of Packages

- Modularity: Encapsulate related constructs.
- Easier application design: Code and compile specification and body separately.
- Hiding information:
 - Only the declarations in the package specification are visible and accessible to applications.
- Private constructs in the package body are hidden and inaccessible.
- All coding is hidden in the package body.



Advantages of Packages

- Persistency of variables and cursors
- Better performance
 The entire package is loaded into memory when the package is first referenced.
- · There is only one copy in memory for all users.
- The dependency hierarchy is simplified.
- Overloading: Multiple subprograms of the same name



Summary

- Improve organization, management, security, and performance by using packages
- Group related procedures and functions together in a package
- Change a package body without affecting a package specification
- Hide the source code from users
- Load the entire package into memory on the first call
- Reduce disk access for subsequent calls
- Provide identifiers for the user session